Chris Perkins
04/26/2012

Implementation of QT Algorithm for STAR MTD : Run 2012

QT Code Version: 0x69
MCS File: qt32b_l0_v6_9.mcs

Description:

This algorithm outputs the maximum TAC Pair Sum from eight MTD modules and a bitmap of modules that had a TAC Pair Sum within a window of the maximum TAC Pair Sum. Because of limited bit availability between daughter cards, the bitmap may indicate that a module was within the window when in fact it was **not** within the window. However, if the bitmap indicates that a module was **not** within the window, it is guaranteed that the module was actually **not** within the window. As a result of this *approximate* hitmap, an upstream DSM may over-count the actual multiplicity but will never under-count the actual multiplicity. See the included pseudocode to reproduce the hitmap.

Each MTD module has an East and West end, which are connected to either the first two or second two channels (2xADC and 2xTAC) on any QT8 daughter card. The standard mask can be used for each channel to mask that channel from the trigger but retain the data in the datastream. Note that separate masks must be used for ADC and TAC channels.

This algorithm assumes the standard configuration where channels 1-4 are signal inputs and channels 5-8 are TAC inputs corresponding to channels 1-4. It uses the standard “Good Hit” definition, which requires that a given ADC channel is greater than some ADC Threshold and the corresponding TAC channel is greater than some TAC_MIN and less than some TAC_MAX. For a module to be considered for the Max TAC Pair Sum or indicate a hit in the hitmap, both East AND West must satisfy the “Good Hit” requirement for that module. If either the East or West end of a module doesn’t satisfy the “Good Hit” requirement, the TAC Pair Sum for that module will be ‘0’ and the hitmap bit will be ‘0’ even if a TAC Pair Sum of ‘0’ falls within the window. If no module satisfies the “Good Hit” requirement, the Max TAC Pair Sum will be ‘0’.

Inputs:

QT8A :

Ch 1/2 : MTD Module 1 East/West ADC
Ch 3/4 : MTD Module 2 East/West ADC
Ch 5/6 : MTD Module 1 East/West TAC
Ch 7/8 : MTD Module 2 East/West TAC

QT8B :

Ch 1/2 : MTD Module 3 East/West ADC
Ch 3/4 : MTD Module 4 East/West ADC
Ch 5/6 : MTD Module 3 East/West TAC
Ch 7/8 : MTD Module 4 East/West TAC

QT8C :

Ch 1/2 : MTD Module 5 East/West ADC
Ch 3/4 : MTD Module 6 East/West ADC
Ch 5/6 : MTD Module 5 East/West TAC
Ch 7/8 : MTD Module 6 East/West TAC

QT8D :

Ch 1/2 : MTD Module 7 East/West ADC
Ch 3/4 : MTD Module 8 East/West ADC
Ch 5/6 : MTD Module 7 East/West TAC
Ch 7/8 : MTD Module 8 East/West TAC

Registers (1 Set Per Daughter Card)**:**

Alg. Reg. 0 (Reg 13): "Good Hit" ADC Threshold
Alg. Reg. 1 (Reg 14): "Good Hit" TAC_MIN
Alg. Reg. 2 (Reg 15): "Good Hit" TAC_MAX
Alg. Reg. 3 (Reg 16): Max TAC Pair Sum Window

LUT:

TAC timing adjustment/ADC Pedestal subtraction for each channel

Algorithm Latch: 2

L0 Output to DSM:

(0-12) : Maximum TAC Pair Sum (E + W)
(13-20) : Hitmap of modules with TAC Pair Sum within window (Mod1 – Mod8)
(21-31) : '0'

Pseudocode for Hitmap :

Incoming Hitmap Bits (Same algorithm on each Daughter Card) :

```
if (SUM1 > SUM2) then
   if (SUM1 > INCOMING_TAC_SUM_MAX) then
      if ((INCOMING_TAC_SUM_MAX + WINDOW) > SUM1) then
         Pass through incoming hitmap bits.
         (Local SUM1 is the new max but the incoming sum is within the window)
      else
         Zero out incoming hitmap bits.
         (Local SUM1 is the new max and the incoming sum is outside the window)
      end if;
   else
         Pass through incoming hitmap bits.
         (The incoming sum is still the max)
   end if;
else
   if (SUM2 > INCOMING_TAC_SUM_MAX) then
      if ((INCOMING_TAC_SUM_MAX + WINDOW) > SUM2) then
         Pass through incoming hitmap bits.
         (Local SUM2 is the new max but the incoming sum is within the window)
      else
         Zero out incoming hitmap bits.
         (Local SUM2 is the new max and the incoming sum is outside the window)
      end if;
   else
         Pass through incoming hitmap bits.
         (The incoming sum is still the max)
   end if;
end if;
```

Local Hitmap Bits (Same algorithm on each Daughter Card) :

```
if (SUM1 > SUM2) then
   if (SUM1 > INCOMING_TAC_SUM_MAX) then
      TAC_MASK_OUT_L1 <= '1';   (SUM1 is the new max)
      if ((SUM2 + WINDOW) > SUM1) then
         MASK_OUT_L2 <= '1';  (SUM1 is the new max and SUM2 is within window)
      else
         MASK_OUT_L2 <= '0';  (SUM1 is the new max and SUM2 is outside window)
      end if;
   else
      if ((SUM1 + WINDOW) > INCOMING_TAC_SUM_MAX) then
         MASK_OUT_L1 <= '1';  (The incoming sum is still the max but SUM1 is within the window)
      else
         MASK_OUT_L1 <= '0';  (The incoming sum is still the max and SUM1 is outside the window)
      end if;
      if ((SUM2 + WINDOW) > INCOMING_TAC_SUM_MAX) then
         MASK_OUT_L2 <= '1';  (The incoming sum is still the max but SUM2 is within the window)
      else
         MASK_OUT_L2 <= '0';  (The incoming sum is still the max and SUM2 is outside the window)
      end if;
   end if;
else
   if (SUM2 > INCOMING_TAC_SUM_MAX) then
      TAC_MASK_OUT_L2 <= '1';  (SUM2 is the new max)
      if ((SUM1 + WINDOW) > SUM2) then
         MASK_OUT_L1 <= '1';  (SUM2 is the new max and SUM1 is within the window)
      else
         MASK_OUT_L1 <= '0';  (SUM2 is the new max and SUM1 is outside the window)
      end if;
   else
      if ((SUM1 + WINDOW) > INCOMING_TAC_SUM_MAX) then
         MASK_OUT_L1 <= '1';  (The incoming sum is still the max but SUM1 is within the window)
      else
         MASK_OUT_L1 <= '0';  (The incoming sum is still the max and SUM1 is outside the window)
      end if;
      if ((SUM2 + WINDOW) > INCOMING_TAC_SUM_MAX) then
         MASK_OUT_L2 <= '1';  (The incoming sum is still the max and SUM2 is within the window)
      else
         MASK_OUT_L2 <= '0';  (The incoming sum is still the max and SUM2 is outside the window)
      end if;
   end if;
end if;
```

Actions:

Tick	QT8A	QT8B	QT8C	QT8D
1	Mask Channels / **Latch Inputs**	Mask Channels / **Latch Inputs**	Mask Channels / **Latch Inputs**	Mask Channels / **Latch Inputs**
2	ADC > R0 → **ADC_GOOD** TAC > R1 → **TAC_MIN_GOOD** TAC < R2 →	ADC > R0 → **ADC_GOOD** TAC > R1 → **TAC_MIN_GOOD** TAC < R2 → **TAC_MAX_GOOD**	ADC > R0 → **ADC_GOOD** TAC > R1 → **TAC_MIN_GOOD** TAC < R2 → **TAC_MAX_GOOD**	ADC > R0 → **ADC_GOOD** TAC > R1 → **TAC_MIN_GOOD** TAC < R2 → **TAC_MAX_GOOD**
3	**Latch Good Hits**	**Latch Good Hits**	**Latch Good Hits**	**Latch Good Hits**
4	Check Good Hits → **Sum1**, **Sum2**	Check Good Hits → **Sum1**, **Sum2**	Check Good Hits → **Sum1**, **Sum2**	Check Good Hits → **Sum1**, **Sum2**
5	Max(Sum1, Sum2, SUM_IN) → **SUM_OUT** Hitmap_Alg(Sum1, Sum2, SUM_IN, HIT_MASK_IN, R3) → **HIT_MASK_OUT**	Sums → **Sums_Del1**	Sums → **Sums_Del1**	Sums → **Sums_Del1**
6	**Latch out SUM_OUT** **Latch out HIT_MASK_OUT**	Sums_Del1 → **Sums_Del2**	Sums_Del1 → **Sums_Del2**	Sums_Del1 → **Sums_Del2**
7	-	Sums_Del2 → **Sums_Del3** Latch in SUM_IN, HIT_MASK_IN	Sums_Del2 → **Sums_Del3**	Sums_Del2 → **Sums_Del3**
8	-	Max(Sum1_Del3, Sum2_Del3, SUM_IN) → **SUM_OUT** Hitmap_Alg(Sum1_Del3, Sum2_Del3, SUM_IN, HIT_MASK_IN, R3) → **HIT_MASK_OUT**	Sums_Del3 → **Sums_Del4**	Sums_Del3 → **Sums_Del4**
9	-	**Latch out SUM_OUT** **Latch out HIT_MASK_OUT**	Sums_Del4 → **Sums_Del5**	Sums_Del4 → **Sums_Del5**
10	-	-	Sums_Del5 → **Sums_Del6** Latch in SUM_IN, HIT_MASK_IN	Sums_Del5 → **Sums_Del6**
11	-	-	Max(Sum1_Del6, Sum2_Del6, SUM_IN) → **SUM_OUT** Hitmap_Alg(Sum1_Del6, Sum2_Del6, SUM_IN, HIT_MASK_IN, R3) → **HIT_MASK_OUT**	Sums_Del6 → **Sums_Del7**
12	-	-	**Latch out SUM_OUT** **Latch out HIT_MASK_OUT**	Sums_Del7 → **Sums_Del8**
13	-	-	-	Sums_Del8 → **Sums_Del9** Latch in SUM_IN, HIT_MASK_IN
14	-	-	-	Max(Sum1_Del9, Sum2_Del9, SUM_IN) → **SUM_OUT** Hitmap_Alg(Sum1_Del9, Sum2_Del9, SUM_IN, HIT_MASK_IN, R3) → **HIT_MASK_OUT**
15	-	-	-	**Latch out SUM_OUT** **Latch out HIT_MASK_OUT**